

00/62/60
JC922 U.S. PTO

10-04-01

A

09/29/00
JC916 U.S. PTO
09/29/00
JC916 U.S. PTO
09/29/00

BEA9-2000-0007-US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application Transmittal

Commissioner for Patents
Washington, D.C. 20231

Sir:

Transmitted herewith for filing is the Patent Application of:

Inventor: **Douglas R. Miller**

Entitled: **Method For Eliminating Or Reducing Hang Conditions In Computer Systems**

Enclosed are:

14 pages of specifications, claims and abstract
8 sheets of drawings

A Declaration and Power of Attorney

An Assignment of the invention to International Business Machines Corporation, Armonk, New York 10504 and Recordation Cover Sheet form PTO-1595

An Information Disclosure Statement, form PTO-1449, and copies of references

The filing fee has been calculated as follows:

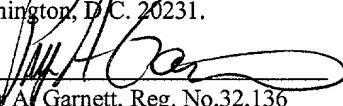
Other Than Small Entity

For:	No. Filed	No. Extra	Rate	Fee
Basic Fee				\$690.00
Total Claims	29 -20 =	9	x \$18.00=	\$162.00
Indep. Claims	2 -3 =	0	x \$78.00=	\$0.00
Multiple Dependent Claim Presented			\$260.00	\$0.00
TOTAL				\$852.00

EXPRESS MAIL CERTIFICATE

Express Mail Label No.: EES19921999US
Date: September 29, 2000

I hereby certify that I am depositing the enclosed or attached paper with the U.S. Postal Service "Express Mail Post Office to Addressee" service on the above date, addressed to the Assistant Commissioner for Patents, Box Patent Application, Washington, D.C. 20231.


Pryor A. Garnett, Reg. No.32,136

Deposit Account Authorization:

Please charge Deposit Account No. 501336 in the amount of \$852.00. A duplicate copy of this sheet is enclosed.

The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit account 501336.

 A duplicate copy of this sheet is enclosed.

Any additional filing fees required under 37 C.F.R. §1.16.

Any patent application processing fees under 37 C.F.R. §1.17.

Respectfully submitted,

By 
Pryor A. Garnett
Attorney for Applicant
Registration No. 32,136

IBM Corporation
Intellectual Property Law Dept., EDO2-805
15450 SW Koll Parkway
Beaverton, OR 97006-6063

(503) 578-5020 voice
(503) 578-5040 fax

METHOD FOR ELIMINATING OR REDUCING HANG CONDITIONS IN COMPUTER SYSTEMS

5

BACKGROUND OF THE INVENTION

10

Technical Field

This invention relates to a method for eliminating or reducing hangs during a shutdown or a panic of computer systems and improves the validity of debugging data acquired from the panic shutdown.

15

Description of the Prior Art

Arguably the most important part of a computer system is the data stored on it. This data is almost exclusively stored on offline storage devices such as disks. It seems to be a continuing trend that the speed of computers remains ahead of the speed of offline storage, and for these reasons most implementations of computer systems use a caching mechanism to keep frequently/recently accessed data in memory rather than on disk. There is a risk with this caching, namely that if anything prevents the cached data from being written to offline storage, the data will be lost. As an example, a sudden loss of power can result in cached data being lost. Another source of cache loss is a failure of the operating system (OS) that results in sudden shutdown without flushing (writing to offline storage) of the caches.

20

Symmetric Multi-Processor (SMP) computer systems use mutual exclusion (mutex) mechanisms to coordinate (prevent collision) of the activities being performed by the various processors (threads of execution). The integrity of these mutexes must be maintained, or severe problems will result including data corruption. Additionally, to prevent problems such as deadlock (system hangs) or system slowdown, it must be ensured that threads which acquire mutexes, do so for a minimal time and according to well-defined rules set forth by the designers of the code. During normal system operation, once the code design has been completed and the implementation sufficiently debugged, these mutexes tend to operate quite well. However, there are some cases where the operation of these mutexes can be disrupted, most notably during system shutdown but also as a result of “bugs” (in both hardware and software), and in general instances of system failure. Typically, the most reliable mechanism for recovering from these disruptions, is to shutdown the system and restart it from scratch.

25

Panic is a special case of OS shutdown. During a normal, orderly shutdown, the OS has the luxury of gracefully terminating applications and subsystems, including database systems and file systems, which results in complete data preservation (i.e. zero data loss or corruption). By

30

35

definition, a panic is the expedited shutdown of the OS and system after detecting a failure of hardware or software. This creates a hostile environment in which the panic shutdown must operate. There are two main goals of a panic:

- 5 1. Minimize corruption and data loss, mainly by synchronization of in-memory (cached) data with off-line storage (disks).
2. Accurately preserve the state of the software and hardware in order to facilitate diagnosis of the root cause of the problem.

10 Note that these two goals may be in conflict under certain circumstances, for example, if a panic is the result of a failure in the disk subsystem. In fact, some circumstances may make it impossible to accomplish either, or both, of these goals.

15 Another intent of a panic is to restart the system with the hope that a fresh start will allow the system to run more successfully. To this end, the time taken by the panic shutdown, core dump, and reboot is critical to the availability of the system and must be minimized. With the unpredictable nature of the system during a panic, there are many impediments to completing the shutdown that may be encountered. One significant impediment is due to the fact that applications have not terminated (gracefully) and thus have not released the mutexes that they may have been holding. The panic shutdown must resolve these mutexes without adding to data corruption, more specifically it must ensure that “dead” mutexes are broken promptly while mutexes held by legitimately running threads are not “stolen”.

20 Aahlad, U.S. Patent Number 5,907,675, provides a method for managing deactivation and shutdown of a server. The patent teaches a method for a server to gently exit without actually terminating the server’s clients. There is a lock-up flag that is used to prevent any new clients from joining. The invention implements an orderly and predictable server deactivation and/or shutdown. The invention utilizes an “usher” that continuously maintains a transaction counter indicative of the number of clients that are actively utilizing services.

25 Hapner, U.S. Patent Number 5,940,827, provides a method for using mutexes to protect the shutdown time commit phase of a database and the shared database cache. The system informs the shutdown thread that the time commit thread is going away. The mutex is created to correspond to a piece of code.

30 However, while Aahlad and Hapner disclose shutting down an application, the present invention relates to shutting down the operating system that supports these applications. The difference between the prior art and the present invention is analogous to turning off one’s television set verses shutting off the power for the whole house.

35 Therefore, there remains an unmet need for expediting panic shutdowns by eliminating or reducing hangs while minimizing data corruption/loss and processing system state information.

SUMMARY OF THE INVENTION

One aspect of this invention is an improved mechanism for dealing with mutual exclusion (mutex) structures in a computer system which can have at least one possible state transition. The mechanism includes an identifier, an indicator, and a mutex handler. The identifier identifies the owner of the mutex, which is preferably a processor or process. The indicator shows whether the mutex was acquired before or after the state transition. Where the state transition is from normal operation of the computer system to a "panic" shutdown, the indicator shows whether the mutex was acquired pre- or post- panic. The mutex handler is responsive to the identifier and the indicator, and in the preferred embodiment includes first routines for use before the state transition, and second routines for use after the state transition.

Another aspect of the invention is a method for handling a mutex after a state transition in the computer system. The method determines whether the mutex was acquired before or after the state transition, and handles the mutex differently depending upon whether the mutex was acquired before or after the state transition. In a preferred embodiment, the mutex determines from a data structure whether the mutex was acquired before or after the state transition.

Yet another aspect of the invention is an article of manufacture comprising a computer-readable signal bearing medium such as a recordable data storage medium or a modulated carrier signal. Means in the medium determine whether the mutex was acquired before or after a straight transition in the computer system, and handle the mutex differently depending upon whether the mutex was acquired before or after the state transition.

The invention thus adapts to panic shutdowns and other state transitions in a computer system, distinguishing between pre- and post-transition mutexes, and handling them differently so as to minimize corruption and data loss and accurately preserve a system state. Other features and advantages of this invention will become apparent from the following detailed description of the presently preferred embodiment, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a 4-node server system according to the invention.

FIG. 2 is flow diagram of a spin mutex used in the system of FIG. 1.

FIG. 3 is flow diagram of a sleep mutex.

FIG. 4 is a flow diagram that depicts panic handling in the system of FIG. 1.

FIGS. 5 and 6 are flow diagrams that depict mutex handling in the system of FIG. 1.

FIG. 7 shows a computer software program product embodying the invention.

FIGS. 8A and 8B show the improved mutex data structure of the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Technical Background

In general, all objects within an OS (or any other software) need a unique identifier (ID), which can be used to refer to the object in an efficient manner. For example, objects stored in an array may use the array index as the ID. Other objects like threads and processes, use a generated ID that is assigned to the object at the time the object is created (or initialized for activity). Processor (engine) objects are also assigned an ID. Since processors (physically) and thread/processes (logically) are actually doing the work of the computer system, these objects/entities can be said to “own” various other objects or entities in the system. When these relationships are being tracked, the objects IDs are used as “owner IDs”.

The present invention includes in the mutex data structure an indicator of whether the mutex was acquired pre-panic or post-panic, preferably by modifying the engineID after the panic is initiated, preferably by assigning the engines different engineIDs post-panic.

The present invention also is a method of using this pre-panic or post-panic indicator during the panic shutdown. The present invention checks the pre-/post-panic indicator (for example, the engine ID) of a mutex, and overrides, waits for, or otherwise handles the mutex differently if it is pre-panic than if it is post-panic.

FIG. 1 shows a representation of a four node non-uniform memory access (NUMA) SMP system 10 with identical replicated OS instances 13. The present invention deals with the panic/shutdown handlers 11 and mutexing (mutual exclusion) primitives 12 of an SMP operating system (OS).

In this example, each processor is represented in the SMP OS 13 with one or more processor-private data structures. In addition, it is assumed that mutexes are implemented in one or more data structures. For the purpose of the present invention, these data structures are extended to include information about the state of the processor and mutex, specifically whether or not the system is in a panic/shutdown state.

Spin Mutex

5 FIG. 2 depicts a spin mutex: A spin mutex is a mutex whose wait mechanism involves simple looping without any throttling mechanism. A processor that is attempting to acquire a spin mutex that is held by another will consume the maximum amount of processor time spinning in a tight loop while repeatedly checking if the mutex is free.

10 Referring to FIG. 2, when attempting to acquire a spin mutex (p_mutex_spin) 21, a tight loop is entered whose only exit criteria is the mutex being free (available) 22. Once the exit criteria are met, the mutex is acquired (held) 23 and the identifier of the processor/thread/process (who acquired the mutex) is stored in the mutex data structure 24. When releasing a spin mutex (v_mutex_spin) 25, the mutex is simply set to a state of “free” (i.e. available) 26.

Sleep Mutex

15 FIG. 3 depicts a sleep mutex. An integral part of acquiring a sleep mutex is the sleeping, which generally implies a potential change of running processor. The pertinent information for the mutex is the processor ID of the processor that acquired the mutex, not the processor that might be running that thread at any given time. Therefore, the processor ID should be used as the mutex ID, set at the time that the mutex was acquired. The purpose of storing this information in the mutex is to provide a target for shoot-down in case this mutex is found to be held after the system enters the post-panic state. It is an implementation choice to either track the processor that acquired the mutex or the processor that last ran the thread holding the mutex. This choice drives whether to use the processor ID or the thread ID. Since the thread ID is always useful for debugging purposes, it may be practical to store both.

20 As shown in FIG. 8A, the processor ID is preferably an eight bit value corresponding to the number of the processor in the system. The processor ID is modified to create the owner ID 72 of the mutex data structure 70 by flipping the high order bit of the processor ID. Thus, processor number five would be assigned processor ID 00000101 (the binary value equal to the decimal number “five”), and the corresponding mutex owner ID 72 would be 10000101. The mutex data structure 70 also

contains additional parameters 74 specific to the type of mutex represented, but those parameters are not pertinent to the invention and are not described further.

When attempting to acquire a sleep mutex (p_mutex_sleep) 31, a loop is entered whose only exit criteria is the mutex being free (available) 32. However, unlike the spin mutex, after each test of the mutex for availability, the thread is put to sleep 33. The only way to be awaked from this sleep is for another thread to release the mutex. This means that the processor running the p_mutex_sleep is capable of running some other thread while this thread waits for the mutex. Once the exit criteria are met, the mutex is acquired 34 and the processor ID identifying the processor/thread/process (who acquired the mutex) is modified 35 to create the mutex ID 72 which identifies the owner of the mutex, and is stored 35 in the mutex data structure 70. When releasing a sleep mutex (v_mutex_sleep) 36, the mutex is set to a state of “free” (i.e. available) 37 and any threads sleeping for this mutex are awakened (variations include waking all or only one of the sleepers) 38.

Various hybrids of spin and sleep mutexes are possible, such as spinning for a period of time and then sleeping. Some variations use a time-delay instead of sleep, or in addition to it.

Note that both FIGS. 2 and 3 depict only general actions. Typically, much if not all of the above will have to be done atomically. In addition, there may be changes made to interrupt priority levels, etc. How this is accomplished is platform dependent. Additionally, there may be sanity checking such as self-deadlock, hang (timeout), and stolen mutex detection. All of these are well known and well within the skills of one having ordinary skill in these arts.

Panic Handling of a Shutdown

FIG. 4 shows the state transition of a general system shutdown implementing this invention, more specifically, a panic shutdown. Although both panic and "voluntary" shutdowns are handled similarly, with respect to mutexes, only the panic case is described for illustration purposes.

Once a panic is initiated 41, the panicking processor changes its flag to indicate that it is legitimately operating in the panic state 42, and changes the mutex routines used by the system to be the ones for panic 43. An additional task is the setting of some global system state or flag to indicate that the system is panicking.

5 Other standard panic activities might include shooting down all other processors 44, general clean up of the system 45 (possibly including flushing buffers and generating a core dump), and then returning to firmware control 46 which may initiate a reboot, or simply halt or power-off.

10

Mutex Handling, Normal vs. Panic

FIG. 5 depicts mutex handling implementing this invention. During normal system operation, acquiring a mutex 51 consists of the normal process for that mutex, plus the setting of a mutex flag 52 to indicate the mutex was acquired while the system was in the "normal" state.

During a panic 54, acquiring a mutex consists of checking if the running processor has been transitioned to the panic state ("does the processor object have its panic flag set?") 55 and if not, then jumping directly to the processor shutdown code 56, and checking the mutex for having been acquired in the "Normal" system state 58 in which case the processor that holds the mutex is shot-down 59 (perhaps redundantly) and the mutex is forced to be free 60. Once a mutex is acquired 61, the mutex flag is set to indicate that it was acquired during the "Panic" system state 62. This indicator of a mutex's post-panic acquisition is accomplished by modifying the identifier of the owner of the mutex.

25 As seen in FIG. 8B, a post-panic mutex's mutex ID 72' is different than the identifier 72 of a pre-panic mutex, in that the two high-order bits of the pre-panic mutex ID 72 are flipped. Thus, a mutex owned by processor number 5 would be given a pre-panic mutex ID 72 equal to 10000101 as described above and shown in FIG. 8A, but would be give a post-panic mutex ID 72' equal to 01000101 as shown in FIG. 8B. The other parameters 74 of the post-panic mutex data structure 70' are not changed from their pre-panic values. Of course, one of ordinary skill will understand that various other ways

30

exist to include a pre-/post-panic indicator in a mutex data structure, or in an external data structure referenced by the mutex handling routines.

Referring now to FIG. 6 during normal system operation 63, releasing a mutex 64 consists of the normal process for that mutex. During a panic 65, releasing a mutex 5 consists of checking the running processor has been transitioned to the panic state (does the processor object have its panic flag set?) and if not then jumping directly to the processor shutdown code 67, and the normal process 64 for that mutex. Note however, that while the mutex ID and flag could be cleared, their contents are not valid unless the mutex is held, and so clearing the mutex ID and flag would be unnecessary.

10

Computer Program Product

FIG. 7 shows an article of manufacture or a computer program-product including a signal-bearing-media for storing thereon program means for carrying out the method and for embodying the data structure. While the present invention has been described above in the context of a data structure and methods of operation, the mechanisms of the present invention are capable of being distributed as a program-product in a variety of forms, and the present invention applies equally regardless of the particular type of signal-bearing-media used to actually carry out the distribution. Examples of signal-bearing-media include recordable media (such as magnetic floppy and hard disks, and CD-ROMs, biological data storage materials, and atomic structure data storage) and transmission media (such as digital and analog communications links, including fiber and wireless communications links).

25

An example of such an article of manufacture is illustrated in FIG. 7 as pre-recorded CD-ROM 1002. CD-ROM 1002 is intended for use with a data processing system, and includes optical storage medium 1004, and program means 1006, 1008, 1010, and 1012 recorded thereon. It will be understood that such apparatus and articles of manufacture all fall within the spirit and scope of this invention. Program means 1006, 1008 causes the operating system to create the improved data structure embodied in the mutex 12. Program means 1010, 1012 cause the OS 13 to perform the steps shown in the flow diagrams of FIGS. 2-6.

30

Advantages Over the Prior Art

This invention permits determination of the context that a mutex was acquired under (pre-panic vs. post-panic) which is not available in other mutex mechanisms.

5

Other mechanisms that are used to resolve shutdown mutex problems, have been limited to timeouts, which have inherent problems including having to guess at how long is "too long" to wait for a mutex and how to measure such a time period in an efficient manner at such a low level, in addition to not providing any intelligence about whether the mutex was legitimately acquire post-panic.

10

Alternative Embodiments

It will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without departing from the spirit and scope of the invention. In particular, if a state transition invalidates the threads executing prior to the transition, then the mutexes must also be transitioned (invalidated). If the state transition is such that all threads are notified and voluntarily terminate, then there is no need for mutex handling. However, if the transition is forced (e.g. from normal operation to a panic; from single uses to multi-user mode; between run levels) then it is necessary to properly handle not only the mutexes but also any threads that failed to make the transition. The present invention can therefore be extended to include state transitions of not only processors and systems, but also threads and processes. Accordingly, the scope of protection of this invention is limited only by the following claims and their equivalents.

3320
3325
3330
3335
3340
3345
3350
3355
3360
3365
3370
3375
3380
3385
3390
3395
3400
3405
3410
3415
3420
3425
3430
3435
3440
3445
3450
3455
3460
3465
3470
3475
3480
3485
3490
3495
3500
3505
3510
3515
3520
3525
3530
3535
3540
3545
3550
3555
3560
3565
3570
3575
3580
3585
3590
3595
3600
3605
3610
3615
3620
3625
3630
3635
3640
3645
3650
3655
3660
3665
3670
3675
3680
3685
3690
3695
3700
3705
3710
3715
3720
3725
3730
3735
3740
3745
3750
3755
3760
3765
3770
3775
3780
3785
3790
3795
3800
3805
3810
3815
3820
3825
3830
3835
3840
3845
3850
3855
3860
3865
3870
3875
3880
3885
3890
3895
3900
3905
3910
3915
3920
3925
3930
3935
3940
3945
3950
3955
3960
3965
3970
3975
3980
3985
3990
3995
4000
4005
4010
4015
4020
4025
4030
4035
4040
4045
4050
4055
4060
4065
4070
4075
4080
4085
4090
4095
4100
4105
4110
4115
4120
4125
4130
4135
4140
4145
4150
4155
4160
4165
4170
4175
4180
4185
4190
4195
4200
4205
4210
4215
4220
4225
4230
4235
4240
4245
4250
4255
4260
4265
4270
4275
4280
4285
4290
4295
4300
4305
4310
4315
4320
4325
4330
4335
4340
4345
4350
4355
4360
4365
4370
4375
4380
4385
4390
4395
4400
4405
4410
4415
4420
4425
4430
4435
4440
4445
4450
4455
4460
4465
4470
4475
4480
4485
4490
4495
4500
4505
4510
4515
4520
4525
4530
4535
4540
4545
4550
4555
4560
4565
4570
4575
4580
4585
4590
4595
4600
4605
4610
4615
4620
4625
4630
4635
4640
4645
4650
4655
4660
4665
4670
4675
4680
4685
4690
4695
4700
4705
4710
4715
4720
4725
4730
4735
4740
4745
4750
4755
4760
4765
4770
4775
4780
4785
4790
4795
4800
4805
4810
4815
4820
4825
4830
4835
4840
4845
4850
4855
4860
4865
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
4925
4930
4935
4940
4945
4950
4955
4960
4965
4970
4975
4980
4985
4990
4995
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
5155
5160
5165
5170
5175
5180
5185
5190
5195
5200
5205
5210
5215
5220
5225
5230
5235
5240
5245
5250
5255
5260
5265
5270
5275
5280
5285
5290
5295
5300
5305
5310
5315
5320
5325
5330
5335
5340
5345
5350
5355
5360
5365
5370
5375
5380
5385
5390
5395
5400
5405
5410
5415
5420
5425
5430
5435
5440
5445
5450
5455
5460
5465
5470
5475
5480
5485
5490
5495
5500
5505
5510
5515
5520
5525
5530
5535
5540
5545
5550
5555
5560
5565
5570
5575
5580
5585
5590
5595
5600
5605
5610
5615
5620
5625
5630
5635
5640
5645
5650
5655
5660
5665
5670
5675
5680
5685
5690
5695
5700
5705
5710
5715
5720
5725
5730
5735
5740
5745
5750
5755
5760
5765
5770
5775
5780
5785
5790
5795
5800
5805
5810
5815
5820
5825
5830
5835
5840
5845
5850
5855
5860
5865
5870
5875
5880
5885
5890
5895
5900
5905
5910
5915
5920
5925
5930
5935
5940
5945
5950
5955
5960
5965
5970
5975
5980
5985
5990
5995
6000
6005
6010
6015
6020
6025
6030
6035
6040
6045
6050
6055
6060
6065
6070
6075
6080
6085
6090
6095
6100
6105
6110
6115
6120
6125
6130
6135
6140
6145
6150
6155
6160
6165
6170
6175
6180
6185
6190
6195
6200
6205
6210
6215
6220
6225
6230
6235
6240
6245
6250
6255
6260
6265
6270
6275
6280
6285
6290
6295
6300
6305
6310
6315
6320
6325
6330
6335
6340
6345
6350
6355
6360
6365
6370
6375
6380
6385
6390
6395
6400
6405
6410
6415
6420
6425
6430
6435
6440
6445
6450
6455
6460
6465
6470
6475
6480
6485
6490
6495
6500
6505
6510
6515
6520
6525
6530
6535
6540
6545
6550
6555
6560
6565
6570
6575
6580
6585
6590
6595
6600
6605
6610
6615
6620
6625
6630
6635
6640
6645
6650
6655
6660
6665
6670
6675
6680
6685
6690
6695
6700
6705
6710
6715
6720
6725
6730
6735
6740
6745
6750
6755
6760
6765
6770
6775
6780
6785
6790
6795
6800
6805
6810
6815
6820
6825
6830
6835
6840
6845
6850
6855
6860
6865
6870
6875
6880
6885
6890
6895
6900
6905
6910
6915
6920
6925
6930
6935
6940
6945
6950
6955
6960
6965
6970
6975
6980
6985
6990
6995
7000
7005
7010
7015
7020
7025
7030
7035
7040
7045
7050
7055
7060
7065
7070
7075
7080
7085
7090
7095
7100
7105
7110
7115
7120
7125
7130
7135
7140
7145
7150
7155
7160
7165
7170
7175
7180
7185
7190
7195
7200
7205
7210
7215
7220
7225
7230
7235
7240
7245
7250
7255
7260
7265
7270
7275
7280
7285
7290
7295
7300
7305
7310
7315
7320
7325
7330
7335
7340
7345
7350
7355
7360
7365
7370
7375
7380
7385
7390
7395
7400
7405
7410
7415
7420
7425
7430
7435
7440
7445
7450
7455
7460
7465
7470
7475
7480
7485
7490
7495
7500
7505
7510
7515
7520
7525
7530
7535
7540
7545
7550
7555
7560
7565
7570
7575
7580
7585
7590
7595
7600
7605
7610
7615
7620
7625
7630
7635
7640
7645
7650
7655
7660
7665
7670
7675
7680
7685
7690
7695
7700
7705
7710
7715
7720
7725
7730
7735
7740
7745
7750
7755
7760
7765
7770
7775
7780
7785
7790
7795
7800
7805
7810
7815
7820
7825
7830
7835
7840
7845
7850
7855
7860
7865
7870
7875
7880
7885
7890
7895
7900
7905
7910
7915
7920
7925
7930
7935
7940
7945
7950
7955
7960
7965
7970
7975
7980
7985
7990
7995
8000
8005
8010
8015
8020
8025
8030
8035
8040
8045
8050
8055
8060
8065
8070
8075
8080
8085
8090
8095
8100
8105
8110
8115
8120
8125
8130
8135
8140
8145
8150
8155
8160
8165
8170
8175
8180
8185
8190
8195
8200
8205
8210
8215
8220
8225
8230
8235
8240
8245
8250
8255
8260
8265
8270
8275
8280
8285
8290
8295
8300
8305
8310
8315
8320
8325
8330
8335
8340
8345
8350
8355
8360
8365
8370
8375
8380
8385
8390
8395
8400
8405
8410
8415
8420
8425
8430
8435
8440
8445
8450
8455
8460
8465
8470
8475
8480
8485
8490
8495
8500
8505
8510
8515
8520
8525
8530
8535
8540
8545
8550
8555
8560
8565
8570
8575
8580
8585
8590
8595
8600
8605
8610
8615
8620
8625
8630
8635
8640
8645
8650
8655
8660
8665
8670
8675
8680
8685
8690
8695
8700
8705
8710
8715
8720
8725
8730
8735
8740
8745
8750
8755
8760
8765
8770
8775
8780
8785
8790
8795
8800
8805
8810
8815
8820
8825
8830
8835
8840
8845
8850
8855
8860
8865
8870
8875
8880
8885
8890
8895
8900
8905
8910
8915
8920
8925
8930
8935
8940
8945
8950
8955
8960
8965
8970
8975
8980
8985
8990
8995
9000
9005
9010
9015
9020
9025
9030
9035
9040
9045
9050
9055
9060
9065
9070
9075
9080
9085
9090
9095
9100
9105
9110
9115
9120
9125
9130
9135
9140
9145
9150
9155
9160
9165
9170
9175
9180
9185
9190
9195
9200
9205
9210
9215
9220
9225
9230
9235
9240
9245
9250
9255
9260
9265
9270
9275
9280
9285
9290
9295
9300
9305
9310
9315
9320
9325
9330
9335
9340
9345
9350
9355
9360
9365
9370
9375
9380
9385
9390
9395
9400
9405
9410
9415
9420
9425
9430
9435
9440
9445
9450
9455
9460
9465
9470
9475
9480
9485
9490
9495
9500
9505
9510
9515
9520
9525
9530
9535
9540
9545
9550
9555
9560
9565
9570
9575
9580
9585
9590
9595
9600
9605
9610
9615
9620
9625
9630
9635
9640
9645
9650
9655
9660
9665
9670
9675
9680
9685
9690
9695
9700
9705
9710
9715
9720
9725
9730
9735
9740
9745
9750
9755
9760
9765
9770
9775
9780
9785
9790
9795
9800
9805
9810
9815
9820
9825
9830
9835
9840
9845
9850
9855
9860
9865
9870
9875
9880
9885
9890
9895
9900
9905
9910
9915
9920
9925
9930
9935
9940
9945
9950
9955
9960
9965
9970
9975
9980
9985
9990
9995
10000

CLAIMS

I claim:

1. A mechanism for a mutex in a computer system having at least one possible state transition, comprising:
 - an identifier of the owner of the mutex;
 - an indicator of whether the mutex was acquired before or after the state transition; and
 - a mutex handler responsive to the identifier and the indicator.
- 10 2. The mechanism of claim 1, wherein the state transition is selected from the group consisting of:
 - a state transition between operational states of the computer system;
 - a state transition between operational states of a processor of the computer system;
 - a state transition between states of a process executing in the computer system; and
 - a state transition between states of a thread executing in the computer system.
- 15 3. The mechanism of claim 2, wherein the state transition is between operational states of the computer system.
- 20 4. The mechanism of claim 3, wherein the state transition is from normal operation to a panic.
5. The mechanism of claim 1, wherein the indicator and the identifier are combined and sized to be handled in a single atomic operation.
- 25 6. The mechanism of claim 1, wherein the indicator is a modification of the identifier.
7. The mechanism of claim 1, wherein the identifier identifies a processor.
- 30 8. The mechanism of claim 7, wherein the identifier is a processor ID value.
9. The mechanism of claim 1, wherein:
the identifier is a first value if the mutex is acquired before the state transition; and
the identifier is a second value if the mutex is acquired after the state transition.

10. The mechanism of claim 1, wherein the mutex handler includes:
first mutex handling routines for use before the state transition; and
second mutex handling routines for use after the state transition.

5 11. The mechanism of claim 1, wherein the mutex handler overrides the mutex if the mutex
was acquired before the state transition, and handles the mutex normally if the mutex was
acquired after the state transition.

10 12. A method for handling a mutex after a state transition in a computer system, comprising
the steps of:
determining whether the mutex was acquired before or after the state transition; and
handling the mutex differently depending on whether the mutex was acquired before or
after the state transition.

15 13. The method of claim 12, wherein whether the mutex was acquired before or after the state
transition is determined from a data structure of the mutex.

20 14. The method of claim 12 wherein the step of handling the mutex includes:
overriding the mutex if the mutex was acquired before the state transition; and
handling the mutex normally if the mutex was acquired after the state transition.

25 15. The method of claim 12, further including the step of:
voluntary termination by a thread, where the thread existed before the state transition, and
where the thread accesses the mutex after the state transition.

16. The method of claim 12, further including the step of:
taking additionally steps to terminate the owner of a mutex acquired before the state
transition.

30 17. The method of claim 12, wherein the state transition is selected from the group consisting
of:
a state transition between operational states of the computer system;
a state transition between operational states of a processor of the computer system;
a state transition between states of a process executing in the computer system; and
a state transition between states of a thread executing in the computer system.

35

18. The method of claim 17, wherein the state transition is from a normal operational state of
the computer system to a panic.

19. The method of claim 17, wherein the determining step is responsive to:
5 an identifier of the owner of the mutex; and
an indicator of whether the mutex was acquired before or after the state transition.

20. The method of claim 19, wherein the indicator is a modification of the identifier.

10 21. The method of claim 19, wherein the identifier is a processor ID value.

22. The method of claim 20, wherein:
the identifier is a first value if the mutex is acquired before the state transition; and
the identifier is a second value if the mutex is acquired after the state transition.

23. A computer program comprising computer program code means adapted to perform all
the steps of the method of claim 12 when run on a computer.

24. An article for handling a mutex, comprising:
a computer-readable signal-bearing medium; and
means in the medium for performing the steps of claim 12.

25. An article for handling a mutex after a state transition in a computer system comprising:
a computer-readable signal-bearing medium;
means in the medium for determining whether the mutex was acquired before or after the
state transition; and
means in the medium for handling the mutex differently depending on whether the mutex
was acquired before or after the state transition.

30 26. The article of claim 25, wherein the medium is a recordable data storage medium selected
from the group consisting of magnetic, optical, biological and atomic data storage media.

27. The article of claim 25, wherein the medium is a modulated carrier signal.

28. The article of claim 25, wherein the state transition is selected from the group consisting of:

a state transition between operational states of the computer system;

a state transition between operational states of a processor of the computer system;

a state transition between states of a process executing in the computer system; and

a state transition between states of a thread executing in the computer system.

29. The article of claim 28, wherein the state transition is from a normal operational state of the computer system to a panic.

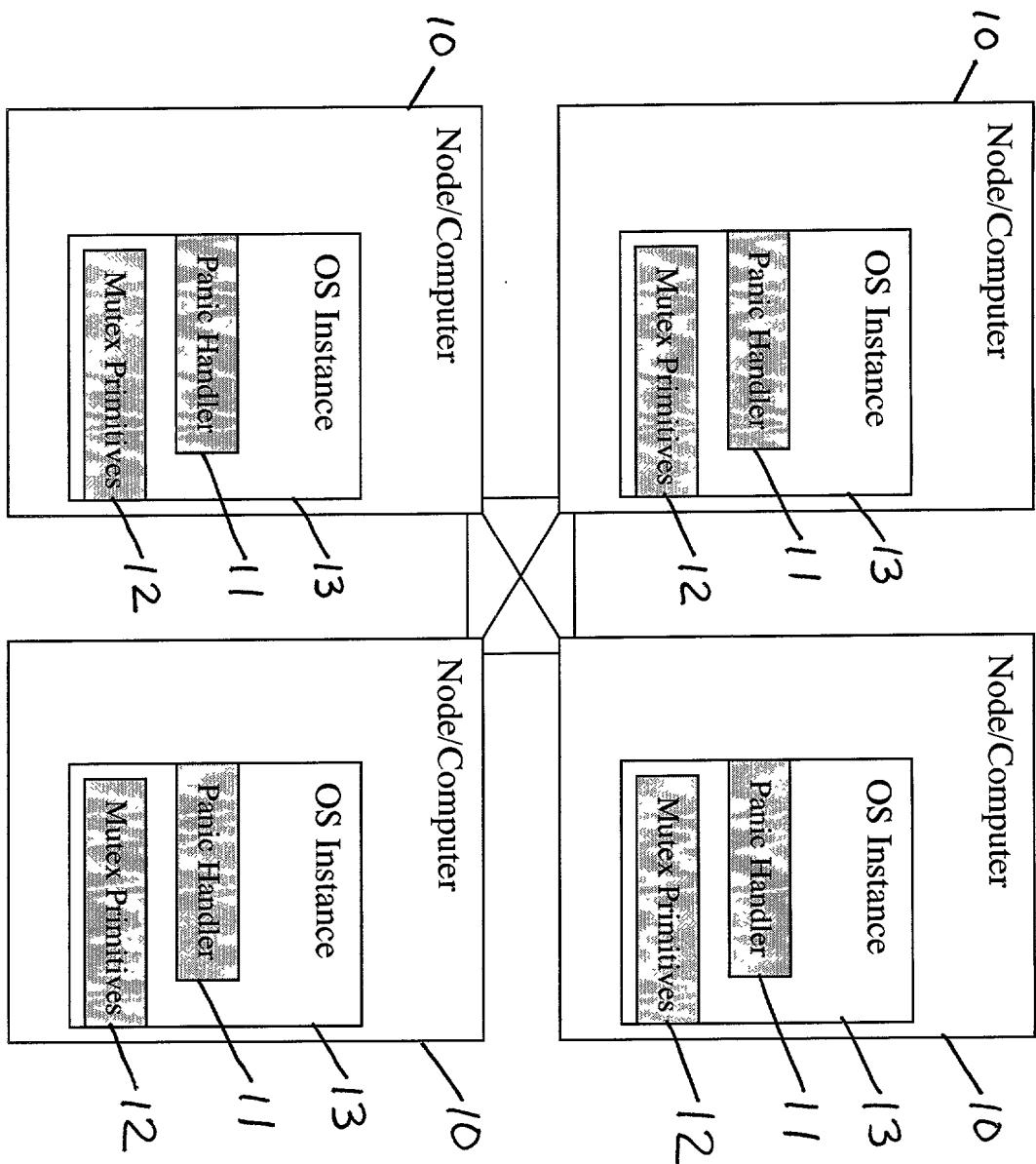
Abstract of the Disclosure**METHOD FOR ELIMINATING OR REDUCING HANG CONDITIONS IN COMPUTER SYSTEMS**

5

An improved data structure handles locks and other mutual exclusion (mutex) mechanisms during a “panic” shutdown of the system such as when the system “hangs”. Existing mutex data structures include an identifier of the engine/processor, the thread, or the processes acquiring the mutex. The improved mutex data structure further includes an indicator of whether the mutex was acquired before or after the panic (pre-panic or post-panic), preferably as a modification of the engineID after the panic is initiated such as by assigning the engines different engineIDs post-panic. The method checks mutexes to determine whether they were acquired pre- or post-panic mutexes. During a panic, alternative mutex handling routines free (release) pre-panic mutexes and shoot down the processors owning these mutexes. The data structure and method are generally useful in state transitions of the system, its engines/processors, and its processes and threads. An article of manufacture embodies the method and data structure in software.

10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95

Fig. 1



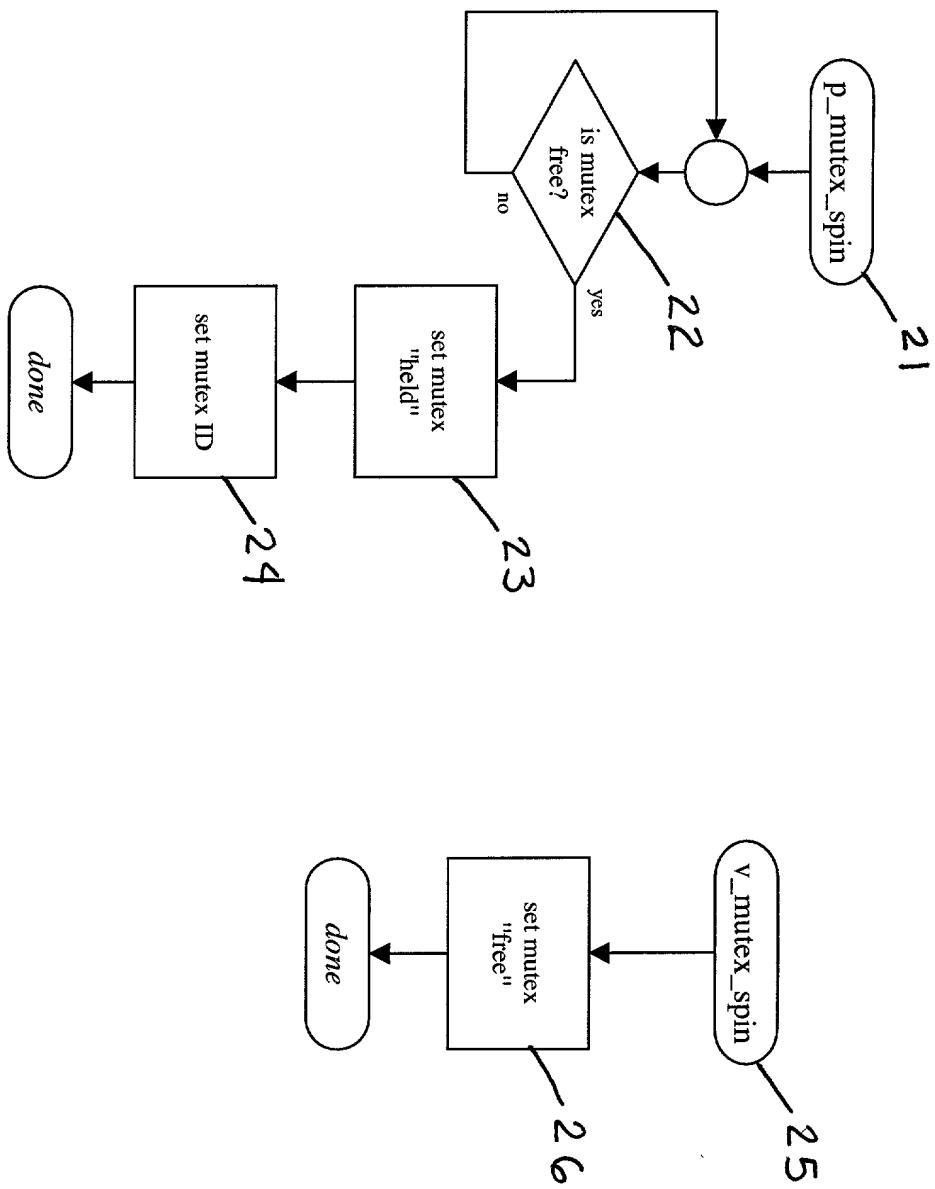


Figure 1: Spin Mutex

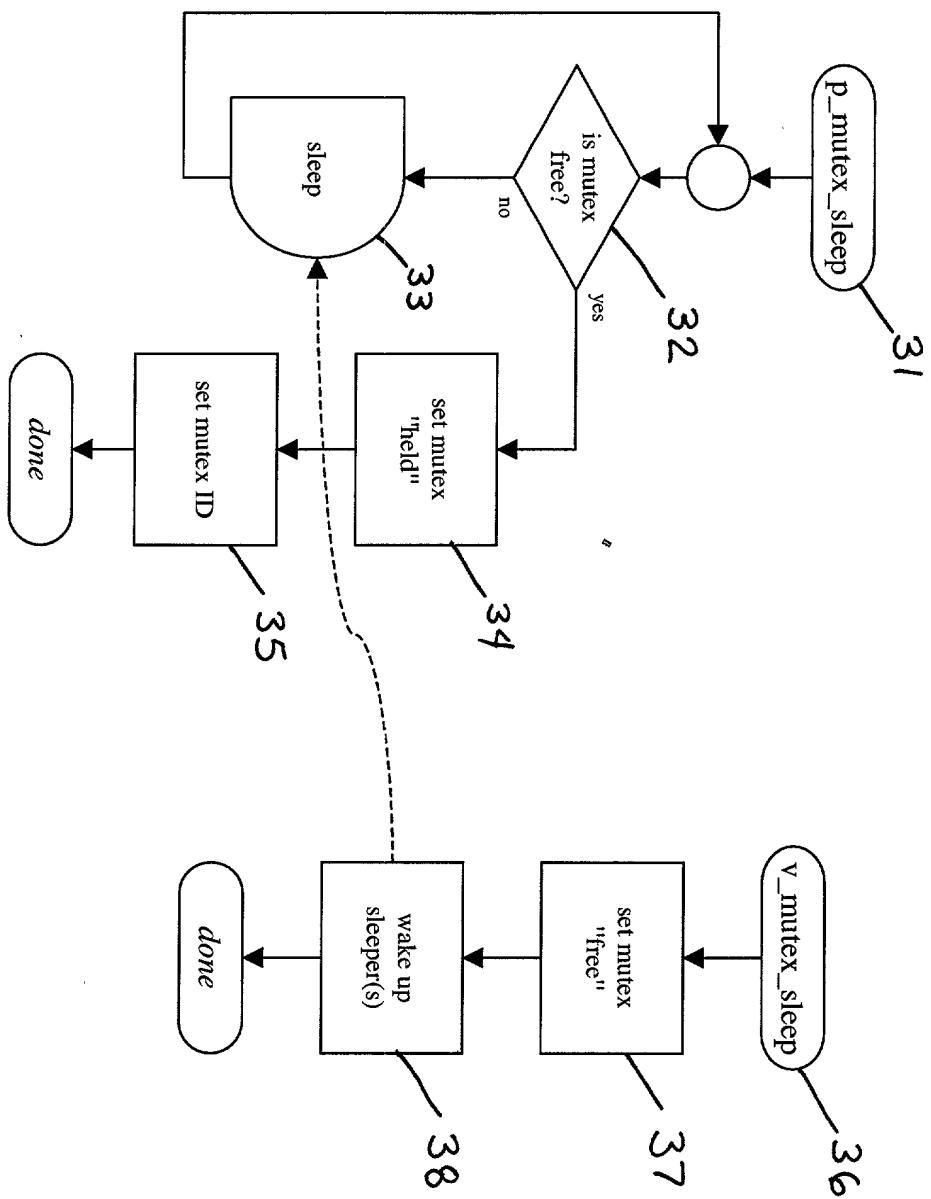


Figure 2a: Sleep Mutex

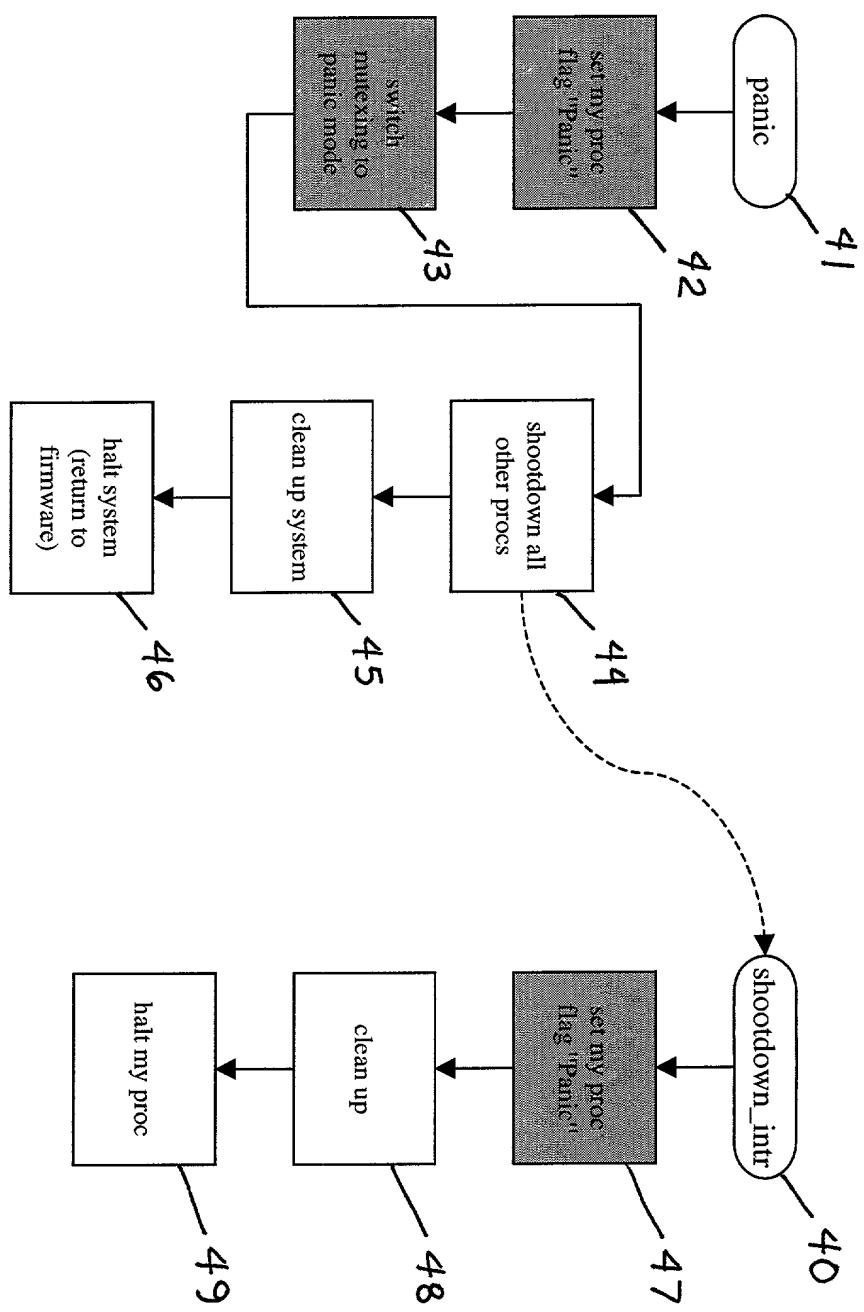


Figure 4: Panic Handling

Figure 4: Mutex Handling, Normal vs. Panic

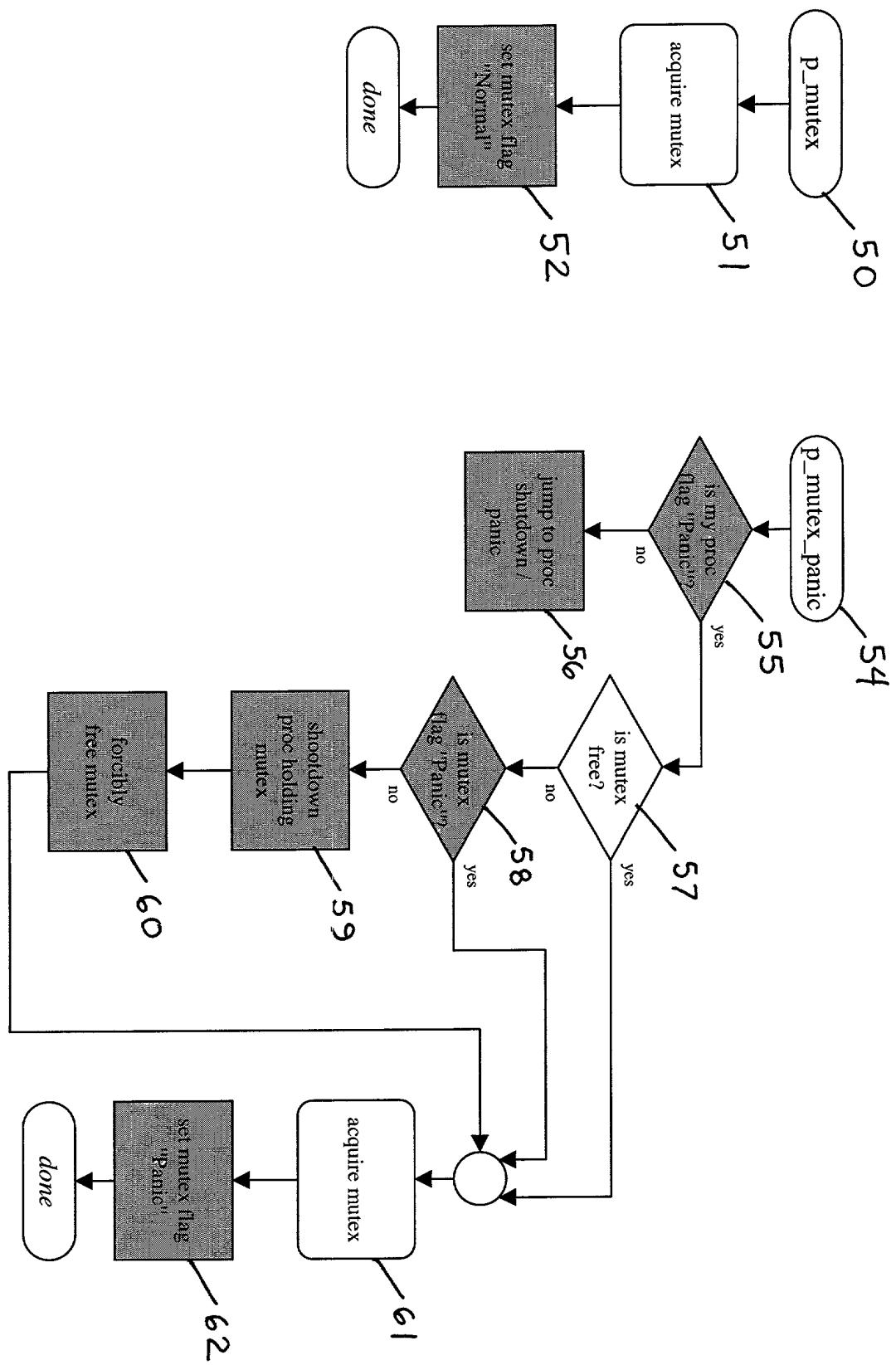
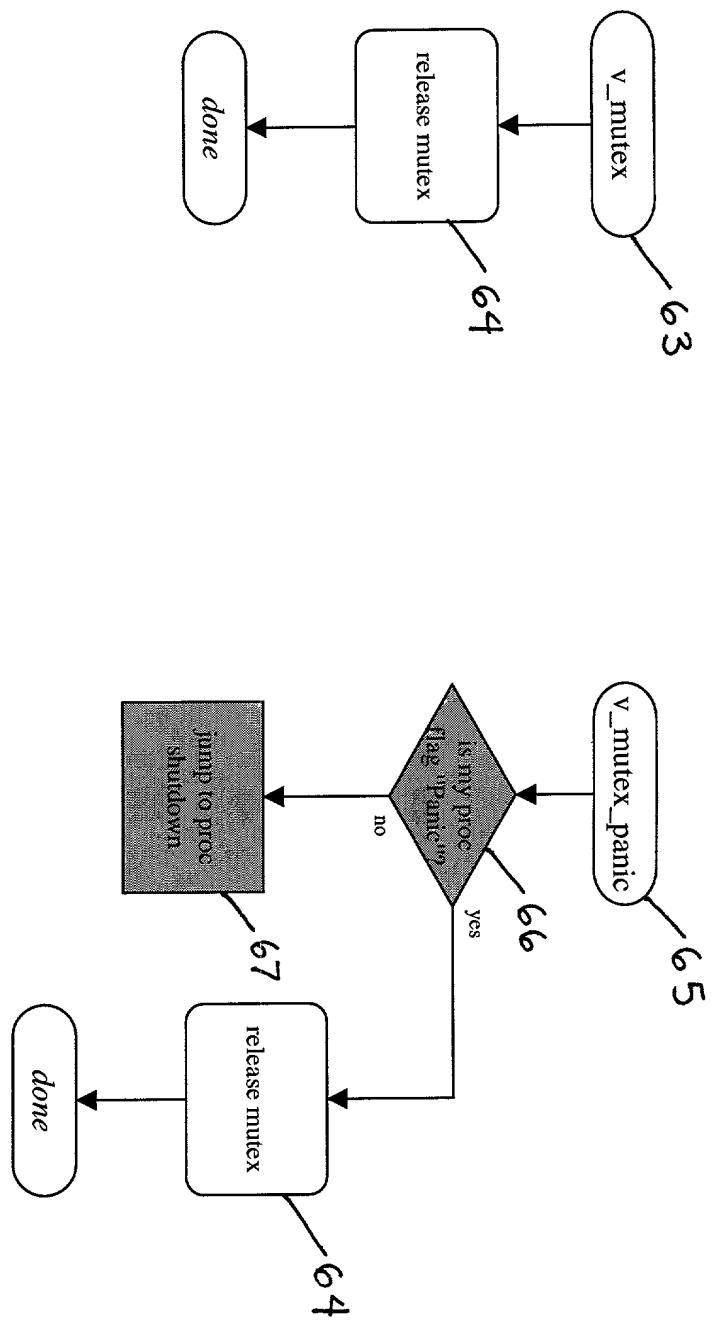


Figure 5: Mutex Handling, Normal vs. Panic (cont.)



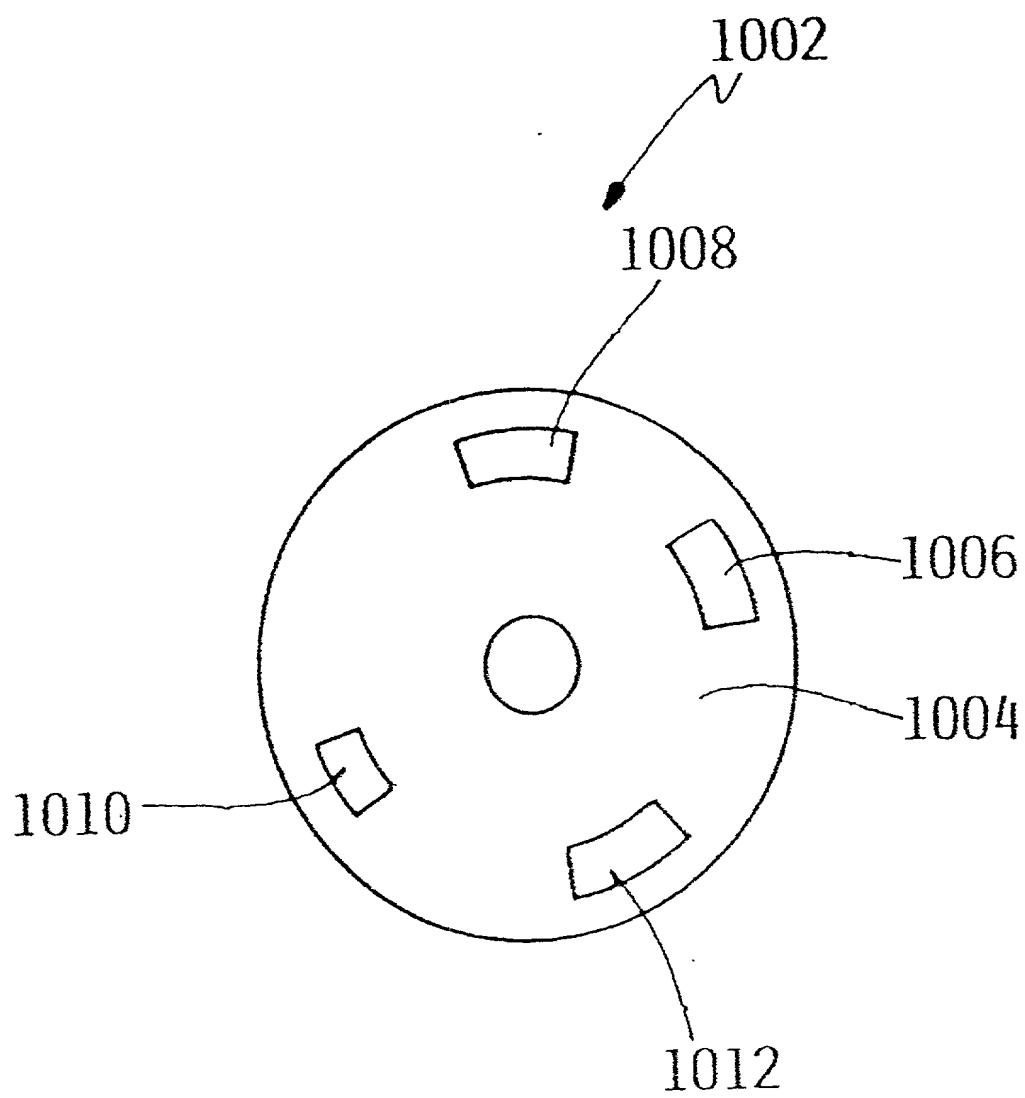


FIG. 7

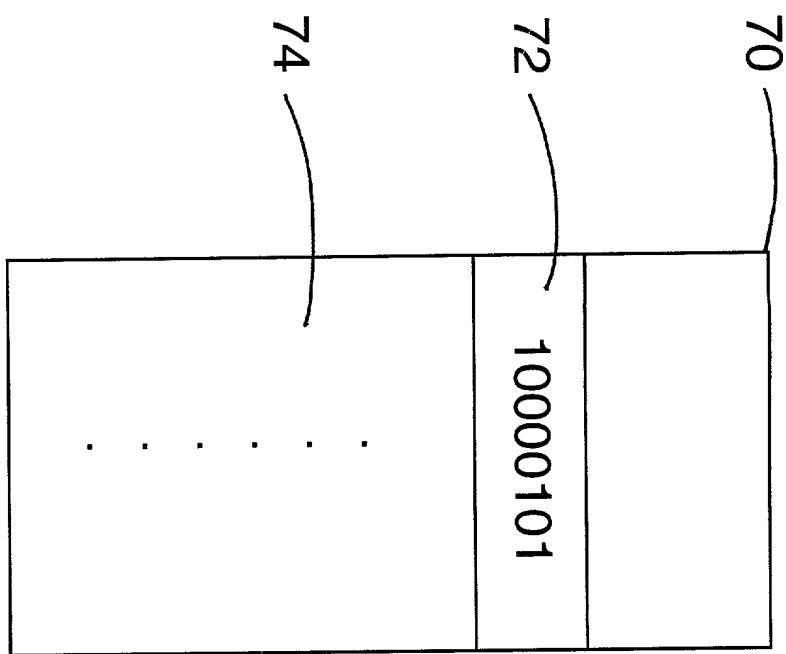


Fig. 8A
Pre-panic mutex

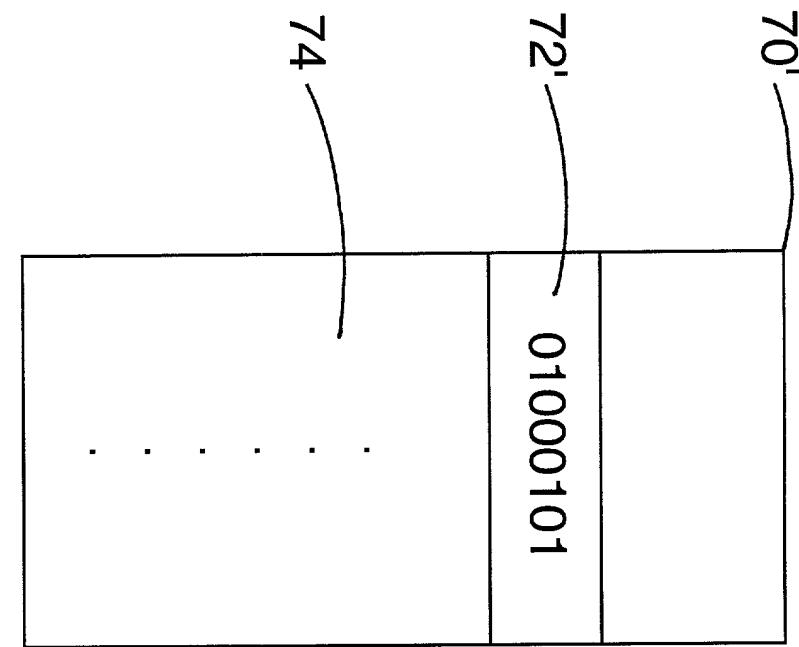


Fig. 8B
Post-panic mutex

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name. I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD FOR ELIMINATING OR REDUCING HANG CONDITIONS IN COMPUTER SYSTEMS

the specification of which is attached hereto.

I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below, and I have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

(NONE)

I claim the benefit under Title 35, United States Code, §120 of any United States Application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(NONE)

All statements made herein of my own knowledge are true, and all statements made on information and belief are believed to be true. Further, these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY

As a named inventor I hereby appoint the following attorneys to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

Pryor A. Garnett - 32,136
William B. Porter, III - 33,135
John E. Hoel - 26,279
Christopher A. Hughes - 26,914
Edward A. Pennington - 32,588
Joseph C. Redmond, Jr. - 18,753
Evelyn H. McConathy - 35,279

Send Correspondence to:

Pryor Garnett, Counsel
IBM Corporation
Intellectual Property Law Dept., EDO2-805
15450 SW Koll Parkway
Beaverton, OR 97006-6063

fax 503-578-5040

Direct Telephone Calls to:

Pryor A. Garnett, 503-578-5020

Full name of sole or first Inventor:

Douglas R. Miller

Douglas Miller
Inventor's signature

29-September, 2000
Date

Residence 19918 NW Gloucester Way
 Hillsboro, OR 97124

Citizenship United States of America

Post Office Address Same as Above